Glitches as a Generative Design Process

Romain Vuillemot, Samuel Huron

Gl¡tch^{e}s as a Generative Design Process

Romain Vuillemot*
Univ Lyon, École Centrale de Lyon, CNRS UMR5205, LIRIS, F-69134, France.

Samuel Huron†
Télécom ParisTech, CNRS i3 (UMR 9217), Université Paris-Saclay

ABSTRACT

Glitches—unexpected coding errors—are often discarded as they are considered as design failures. This is surprising as they relentlessly flourish on the web, due the interest their aesthetic triggers. In this paper, we report a preliminary work that aims at understanding glitches in the context of information visualization. We first conducted an empirical study by collecting glitches examples on social media that we grouped by visual and semantic similarities. Glitches descriptions by their creators allowed us to grasp the interest behind sharing such a failure. However, understanding the (unattended) cause of glitches remains usually difficult as it also remains unknown for their creators; nonetheless we reverse engineered some of them in a synthetic manner. We discuss glitches expressive possibilities, which can be considered as a generative design method for visualization, and suggest including them in design studies as a way to better document design processes.

1 INTRODUCTION

Accidents are an integral part of discovery. Unexpected errors are widespread in science in general (e.g. serendipity [27]) but also in any creation process. Artists rarely produce what they initially intended to do: everything gets changed and shifted from the initial goal. In craft domains, errors have a major impact on the result and the process. For instance, a sculptor could not undo a mark engraved during his sculpting process. Otherwise he has to imagine a way to gain benefit of this error, or to redo his work from scratch or to disguise it. Multiple artists chose to use the unexpected and randomness aspects of the crafting execution as a support of their creation process. Jackson Pollock was painting by moving a dripping paintbrush on top of the canvas. This way, he could express his feeling through an action that can lead to both controlled and unexpected results, more than planning a representation ahead.

Some entire movements in art history actually gained benefits form unexpected and random elements along the creative process: abstract expressionism, surrealism [11], dadaism and cubism [5], among others. After the Second World War, artists applied a similar approach to various medium and domains. Nam June Paik created installations with analogical video Larsen effect: a camera filming its own video output to generate unexpected effects. Dan Graham used the introduction of computers, and digital tools opened up new directions to generate random errors, sometime unexpectedly, and sometimes purposefully.

In computer sciences, a glitch is a code that runs but with a bizarre visual output that is partially or completely random. Glitches can be found in various digital domains which goal is not to generate designs. In video games, glitches are when the game does not behave as programmers wanted to, and allows players to cheat as it provides an unfair advantage. There is actually an activity aimed at finding flaws and exploiting them and referred to as *glitching*. In electronics, glitches occur when some failing electronic device keep having a working visual output that may generate interesting combinations of pixels. Electronics may crash but is often more robust than coding and results in more diverse visual outputs.

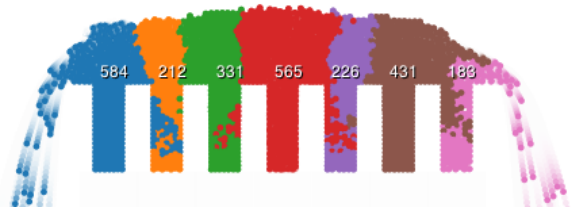

Figure 1: Example of glitch in the parameter space of Visual Sedimentation [13] that resulted in an overflowing bar chart. Such an apparently buggy visualization accually became pointful: the chart shows dramatic increase of populations (individuals are represented as dots and grouped into bars) that have limited space capacity.

The early version of the web (1.0) was also an endless source of glitches due to the medium: slow internet connection, low-resolution and small color palettes on CRT displays, and the design of files format (such as GIF, which are still being used today [17]) allow progressive loading and rendering of images causing some (temporary) distortions. With overall technical improvements of web connections and display systems, glitches became generated either by nostalgia of those times (some examples can be found in the World Wide Wrong exhibition [16] in 2005), to tame anxiety caused by the raising presence of technology [21], or by fiddling with systems that have become too complex [23]. Lately, some artists specifically claimed doing accidental art by generating bugs into digital files [1]. In this movement we can identify different techniques, such as data bending [10] –which is manipulating the data into a media with a text editor in order to generate distortions in the file rendering– playing with compression artifact, and data moshing [6].

However, errors have received little attention in the context of information visualization. Information visualization—the mapping of abstract data to graphical marks and their property, to enhance humans cognition—does not provide an obvious ground for any random process as it is traditionally viewed as an analytical pipeline [14] (Figure 3), which goal is to generate faithful presentations and accurate interpretations by the human. From our experience, we have witnessed interesting randomness in some of our previous work Visual Sedimentation [13] as we visually searched for interesting parameters combination to generate new charts (Figure 1): a simple, standard bar chart becomes interesting and impactful only once it was malfunctioning due to a slight change in its configuration parameters.

It seems errors are actually a fundamental part of visualization, but we are currently only taught to hide and correct them. Expert programmer and visualization designer Mike Bostock stated in [24] *[..] Design has its tendency to be capricious. Which is that some day you fell you are a master of design and things are going pretty*

*e-mail: romain@vuillemot.net
†e-mail: samuel.huron@cybunk.com

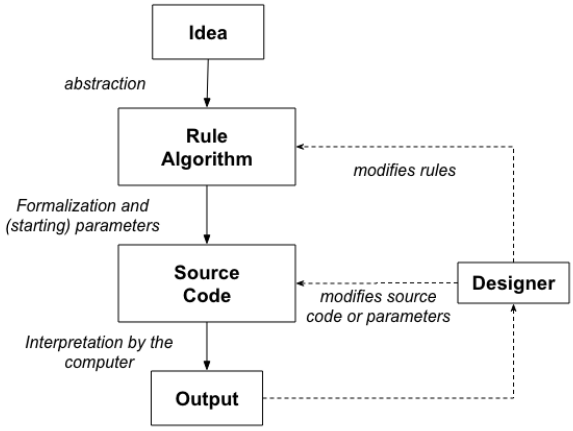

Figure 2: Generating solutions with a parametric design process (adapted from [3]) can be summarized in three steps: choice of parameters; combination of those parameters; validation of the output.

well. And then other days and most days you tend to feel more like a neophyte where you are not really sure what's going wrong how to solve your problems. Why everything is so hard. [..] Quoting a Tweet by Paul Ford that summarizes his sentiments *One commonality between writing and coding is that everything you do is in a nightmarish state of total failure until the moment it is not*. One actually needs to explore things that do not work to find things that work. Mistakes are unavoidable.

This work aims at investigating how unexpected coding errors–glitches–can become an act of discovery rather than only a mistake. As far as we know, this is the very first attempt to explore glitches in the context of information visualization, to provide categories and analyze how they can be combined with creative practices. Such an investigation continues previous efforts that aim at bridging multiple fields together [19], such as arts and visualization, to provide more exchange of ideas.

2 Parametric Design Process and Glitches Generation

Glitches have strong ties with the field of *parametric design* [29, 3] which is dedicated to discover and validate unexpected results from various types of visual or audio outputs, as well as from physical objects. Parametric design finds applications in many fields, from art to genetics, where parameters have been identified to be changed and combined, and finally selected according to an objective criterion. For instance, in architecture, buildings shapes can be defined by a set of variables (height, number of floors, etc.) and their relation. Thus, building shapes variations can be explored by changing those variables, and novel buildings can be selected by criteria such as innovation or optimality [26]. This parametric design process is best summarized by [3] on Figure 2.

Regardless its application domain, parametric design requires first to identify a *parametric space* which refers to all the input variables that can be changed to generate variations. The second step is the *combination of parameters*. This is where randomness plays an important role to fetch permutations that no one has probably thought of beforehand. However, due to the combinatoric of all possible permutations, only a couple of subsets can usually be explored. This is why glitches are particularly interesting: the are only *minor malfunction, mishap, or technical problem* [2], resulting in small and controlled changes, and thus prevents exploring unkown areas. They are only variations of known and (normally) working points in the space. The third and final step is *selection*

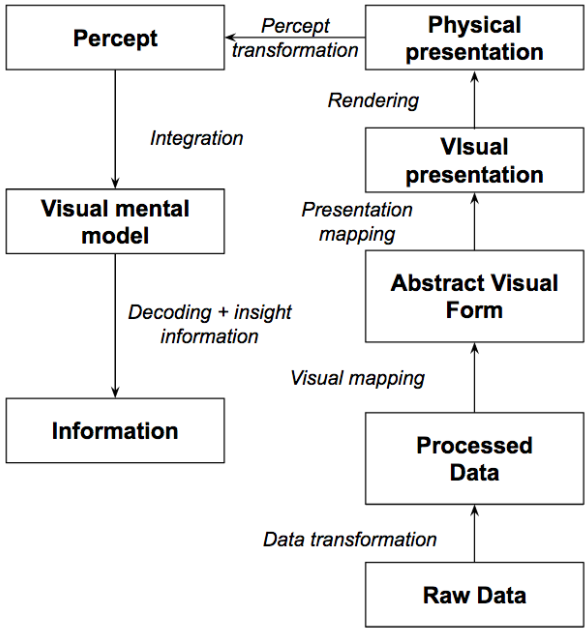

Figure 3: Interaction model (adapted from [14]) provides a list of parameters for visualizations variations.

according to an objective function which allows to consider parameters permutation as valid. The function usually is either qualitative, and can be automated, but can also be quantitative, and then requires human judgement to accept or reject it.

Most of the glitches we presented in the introduction section come from systems that can be described with a parameter space, which captures the complete digital or electronic processing, from capture to perception (e.g. video glitches impact the rendering part of the video, game glitches impact the scene graph of the video game program). In the context of information visualisation, we suggest to use the interaction model proposed in [14] (Figure 3) as parameter space. The model is a series of sequential steps, from *raw data* to *information*, that capture all the transformation and consequently all the possible sources of error. Thus, it provides us with a controlled and limited set of parameters that we can use to understand glitches. This complies with the definition of glitches as being *post-procedural* [22], as a break from a procedural flow. Still, a visualization glitch will have other possible reasons to exist outside this model, and can impact other modalities (audio, etc.). Finally, as most of the visualizations, only a qualitative judgement will tell if the glitch is valid or not in this context.

3 Glitches Collection

To better grasp glitches design space and opportunities for visualization, a naive approach would be to randomly combine all parameters from Figure 3 (e.g. visual mapping of graphical marks, properties, scales, points of view, etc.) and inspect outputs. This approach is not realistic, as it would result in a too important combinatoric problem. Thus we adopt a retrospective approach, by dissecting already created glitches by programmers or artists, in the context of information visualization.

Thankfully, glitches draw increasing scrutiny thanks to web technologies and social media; it is reflected by the growing number of collections available online. Among them, two very active sources of glitches: the first is a Tumblr blog [1] where R [28] developers

[1] http://accidental-art.tumblr.com/archive

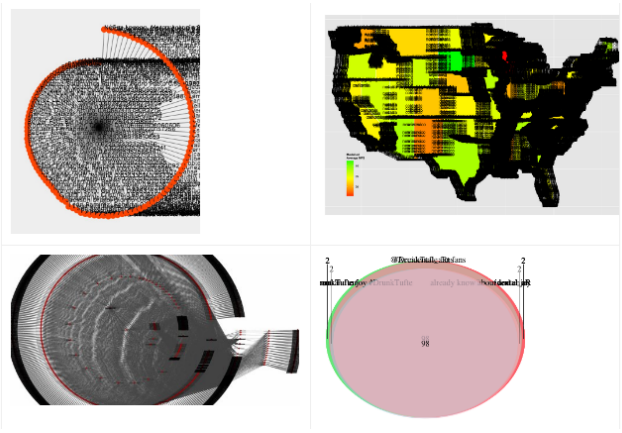

Figure 4: An excessive number of text labels creates new shapes, and emphasizes existing contours.

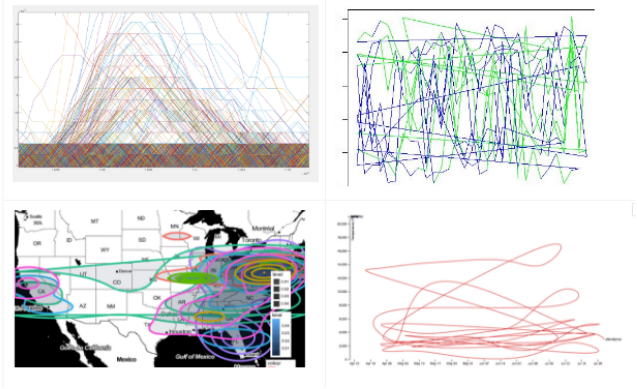

Figure 5: Simple color palettes and scribble-like shapes are considered as glitches as they seem to be hand-drawn by a child.

submit screenshots of their glitches with some comments (intent, dataset, tools); the second is a Twitter hashtag [2] that D3 [4] developers use to also share screenshots along with a Tweet. The Tumblr is curated by moderators, while the Twitter feed is not.

We extracted more than 164 posts from the Tumblr with the oldest one dating back to October 28, 2013. While we were able to extract all the Tumblr posts, the Twitter's search API limited us to programmatically extract 250 from the Twitter hashtag. We ended up with a database of 414 glitches that we did not clean as we wanted to look at all the results in an exploratory manner.The collection and the scripts to create it are available as supplemental material[3].

4 Collection Analysis

We now report our analysis of visualization glitches based on the previously collected samples. This is a *preliminary* and *exploratory* effort due to the lack of comprehensive and structured collection. This led us to use a qualitative approach, rather than quantitative coding, to identify patterns. For each glitch, we asked ourselves simple questions such as what causes the perception of a glitch? Why is the glitch interesting? How the glitch was made? We did our best to answer those questions using the information visualization reference model [14], and inferred authors intents since the rationale behind the glitch was not always available.

[2] #d3brokeandmadeart

[3] https://github.com/romsson/glitches-dataviz-collection

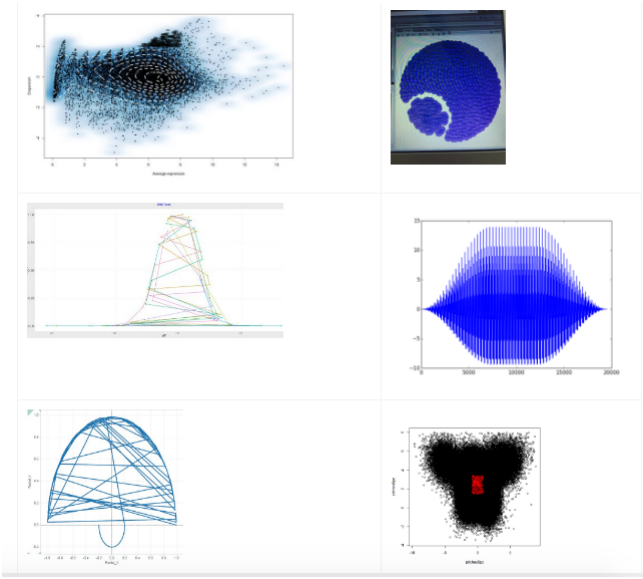

Figure 6: Non visualization-related shapes are perceived instead of a chart, while using the same graphical marks as building blocks.

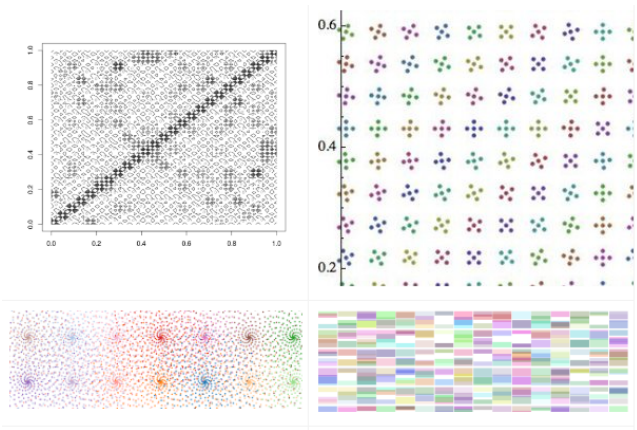

Figure 7: Patterns appear due to the symetrical layouts and well organized graphical elements.

"Text labeling gone wrong"

Many glitches originate due to text labeling errors (Figure 4) by misplacement, overflow or just because too numerous. Such errors occur because labels are attached to *each* graphical mark, instead of only a handful. By consequence, showing all labels at once generates visual black blobs. Those blobs often create an additional layer of graphical marks, which turns the chart in an unexpected shape or emphasizes existing contours. The effect of this particular text glitch is reinforced by our ability to quickly notice and read textual elements, and thus detect any textual randomness immediately. Those glitches also emphasize how labeling is difficult and why interactive techniques, such as eccentric labels [9], should be implemented to only show labels on demand and within a limited perimeter.

"My kids could do that!"

Another frequent cause of glitches is when the result seems too easy (Figure 5). We found charts that creators connected to a children's artwork rather than an actual data-driven plot, and with such sub-

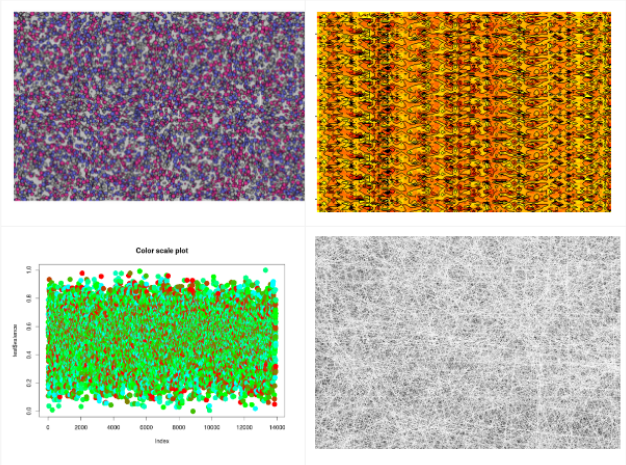

Figure 8: Random noise are a dense display of multiple elements without any particular organization.

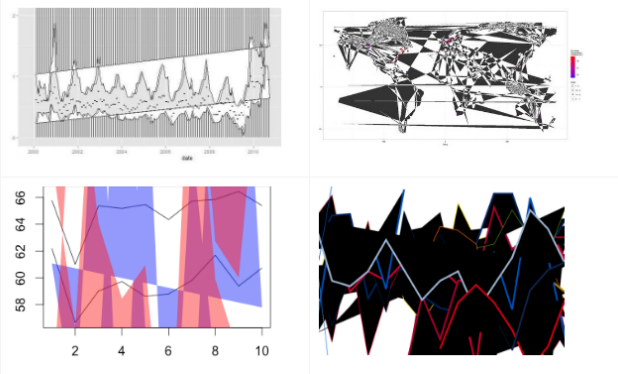

Figure 9: Example of glitches considered as artistic artworks or emblematic of a trend.

titles as *"My kids could do that!"*, *"coulda made it in 10 seconds with a red pen."* or *"A PMT trace when the axes don't cooperate and MATLAB acts like a child with a 64-pack of crayons?"* Despite such glitches are perceived as malfunctioning, they can actually be faithful to the data: a similar and very accurate visualization, such as ZIPScribble maps [18] provides the same reaction. While ZIPScribble connects all ZIP codes in the US, it is locally very random, but some emerging grids show the underlying structure of the dataset. Both ZIPScribble and glitches share the same curious interpretation due to the use of random trajectories combined with simple color palettes.

"I see something"

Some creators, instead of perceiving a visualization, noticed a shape that has a different, non visualization-related meaning (Figure 6). This shape emerged with the same building block as a chart (graphical marks, colors, position), but mentally formed due to its *global* resemblance to a familiar object or subject. *"Accidentally got the Eiffel Tower"*, *"It's a dolphine"*, *"Accidentally drew a moon."*, *"Wanted to make sure an amplitude envelope shaping function was working correctly, so I fed it extreme input arguments, and it gave me a kiss."*, *"Was trying to mess with projections in ggplot. Got a psychadelic doughnut instead! "*, *"It ought to be a social network visualization... Looks like a... flower?"*, *"was trying to overlay some histograms and ended up with something resembling stained-glass windows."*. It sometimes goes in (too much) depth with extra context *"It's supposed to be coefficients of variation by sample mean. Instead it looks like I photocopied my butt."*.

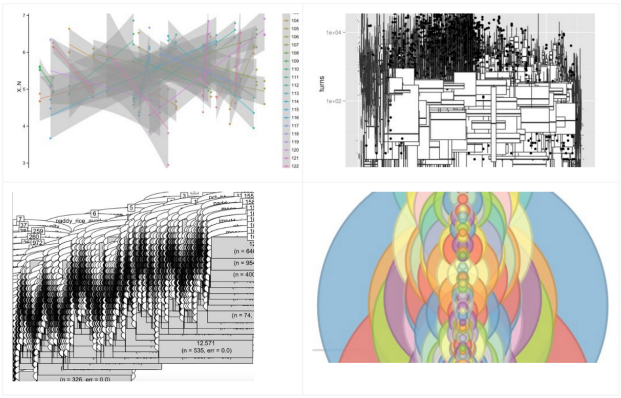

Figure 10: Stacked shapes usually are reported as plain mistakes, instead of artwork or anything able to be identified.

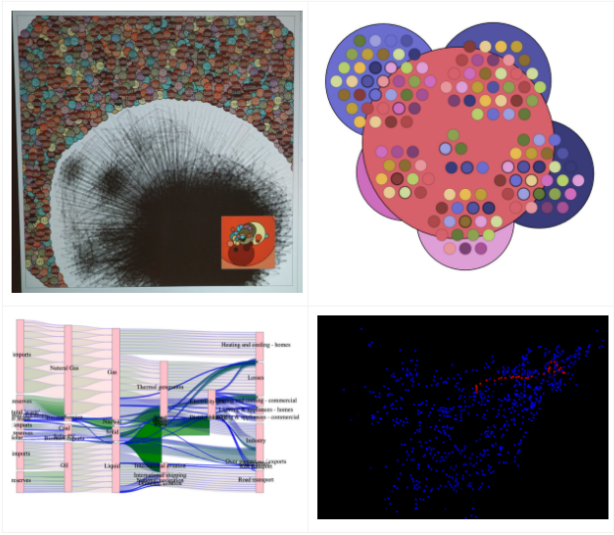

Figure 11: Stories are told by creators to explain relatively complex malformed visual output.

Interestingly, such shapes are relatively far-fetched, but still would not have required any skill to be drawn on purpose. Also, such shapes are relatively non-ambiguous, and are immediately distinguishable. In both cases, the interaction model tells us all those interpretations occur at the end of the pipeline, once the user has perceived the visualization and then mentally decodes it, but in an unexpected way. This was also the case in the previous section were perfectly fine visualizations have turned into a glitch due to its association with some prior and non-related knowledge.

"Patterns and random noise"

On the other side of the visual significance spectrum, creators can be surprised by the apparently non-informative chart, resulting for instance from symetric display of elements (Figure 7) *"Check out those moire patterns!"*, *"Contour plot of random repetitive noise"*, *"I was just fiddling with a colour scale the easiest way I could think of and it produced quite a lovely abstract image."*. Some completely

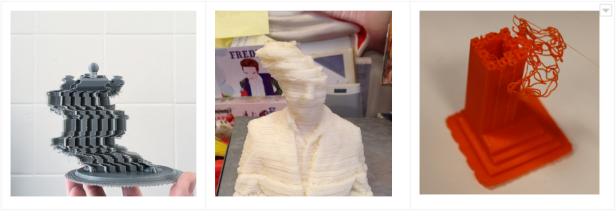

Figure 12: 3D printers can also generate glitches with wrong assembly of physical elements.

random noise patterns (Figure 8) have also been reported but usually without much comment. We assumed such glitches are self-explanatory, while it is still unclear for us why creators are confident that something wrong happened. From a visualization perspective, there is obviously some over plotting and clutter, but creators did not comment any in those terms.

"Artistic reminiscence"

Figure 9 shows glitches considered as artistic artworks or emblematic of a trend. *"Tried to plot points relative to expression level. The output is aRt."*, *"Plotting data with incorrect spatial parameters lead to this very modern piece of art."*, *"a time-series plot led to minimalism in black and blue"* or *"Mistaking 'cex' for 'pch' leads to a nice example of minimalism"*, *"not quite mondrian, but i'm trying"*. There was also additional details on the context in which such art should be displayed *"printed on a 1 x 2 m canvas, "the map" would surely make a nice wall decoration in a Le Corbusier style house."*, *"The bump chart of Dorian Gray, now on display at the Museum of Modern"* or just *"a great desktop image!"*.

It is interesting to note that not all glitches have a high aesthetic value. *"A beautiful error"*, *"little fancier"*, *"fun to look at!"*. Random glitches can also be seen as what they are: random glitches. Many glitches are shapes stacked on top of each others due to issues with layout or opacity (Figure 10), and are explicitly reported as plain mistakes. The frontier between visualization and art is even challenged *"Still unclear of what's art? On the left is #d3brokeandmadeart. On the right is a perfectly reasonable bump chart"*.

"Little stories"

A series of glitches without any particular aesthetic (Figure 11) got some interesting comments: those were little stories such as *"circle pack got infected by one of those protozoa that make snails commit suicide"*, *"It's a blackout in the 19th century."*, *"revenge of the nodes"*, *"According to this chart, we will reach peak dataviz well before peak oil, peak coal and peak hype"*. Those seem to be very personal as they would be difficult to guess without any comment from its creator. Those glitches come from to the author's mental model when conceiving the visualization, rather than objective visual criteria or a shared background. This contrasts with the shapes in a previous section that were very obvious to guess and non-ambiguous.

5 Glitches Reverse Engineering

In this section, we create coding errors–on purpose–which visual results can be assimilated to glitches. This seems to be in conflict with the essence of a glitch, which should be unexpected: *"One does not create a glitch, but triggers it"* [20]. However, our goal is pedagogical to explain which parts of a program may lead to visual disfunctions, without breaking the whole program. We provide code snippets to exemplify simple, yet frequent glitches in the context of D3 [4] programming with `<svg>` rendering.

Inadvertent use of default values. One of the most frequent origins of errors is forgetting to change default values: those values are forced by the system if they have not been set in the code beforehand. Such values are automatically set to guaranty a visual output that works. Those values can be basic (e.g. for color: black, white) or informed (e.g. color scales [12]). Since D3 visualizations often use SVG as the primary rendering mechanism, it relies upon SVG default values. Thus, any plot of elements such as a `<path>` comes with default values for attributes such as color, stroke width, fill, among other attributes [25]. The following code snippet is the correct code to prevent the default `<path>` fill which is black (`fill: black;`) when one wants to draw a line and remove the fill (`fill: none;`). This code should be included in the style sheet of the HTML page. It exemplifies glitches grouped on Figure 9.

```
<style>
/* thin blue line without any area fill */
path {
stroke: steelblue;
stroke-width: 1;
fill: none; /* default is black */
}
</style>
```

Keep drawing while it should have stopped. Most of the unexpected scribble charts on Figure 5 are due to a common error in D3 code structure when using the `.data()` binding attribute. Indeed, the attributes usually requires input data an array of arrays (`[[..], [..], [..]]`) to draw separate lines; While it is a common error to pass only a single flat array (`[.., .., ..]`). The visual result is globally correct, but instead of segments, a continuous `<path>` is drawn. This is very much similar as drawing separate lines on paper, but without lifting the pen between lines.

```
<style>
/* Oringinal dataset [ [1, 2], [3, 4]] */
.data([1, 2, 3, 4])
...
</style>
```

6 More Glitching

There are many other types of coding errors that can lead to glitches, than the ones we have collected, discussed or reverse engineered. As we said earlier, the interaction model [14] can be seen as a parameter space where errors may occur at any transformation step, before and after perception. For instance at the *processed data* and *data filtering* leve, an error can happen by the distortion or the miscoding of the original dataset. This could be due to missing values, outliers and inconsistencies [7], or other factors. At the level of *abstract visual form*, errors appear with the miscoding of the rules of classical chart like changing angles of the bars of a bar chart, or mapping the values to the wrong visual variables. At the level of the *visual presentation*, glitches could appear on labels like the text labeling category. Finally, the *physical presentation* level could also be a place of glitches by distortion of the rendering process such as the way standard glitches are generated with their unique visual aesthetic [22, 8, 16, 17, 5].

Finally, there is also an important source of glitches when visualizations are *physically rendered*. Figure 12 shows that data physicalization [15] is already a source of glitches for 3D printed objects [4]. In general, physical objects provide more organic changes, which can easily be manipulated by humans, and trigger their curiosity. The physical world could be a source of inspiration to identify charts parameters and change them, similar to our earlier Visual

[4] https://www.flickr.com/groups/3d-print-failures/pool/

Sedimentation [13] bar chart example, for the following reason: changing physical parameters always results in an errors that can be interpreted as a glitch. The reason being that the physical world has few parameters, mostly gravity and material properties, and errors usually mean just collapsed or mishaped objects. This contrasts with changing the visualization reference models parameters—such as performing standard data bending—which can easily break everything and result in no output at all.

7 Conclusion and perspectives

We tend to only present polished end results, discarding intermediary steps. This paper shows that such steps–that we referred to as glitches–should be captured, saved and inspected, to become part of the design process. Furthermore, by sharing them, they could open up new creative spaces that would emerge in a collective manner.

From our preliminary observations of multiple glitches collections, we envision a huge educational potential. Indeed, by showing edge cases of visualizations, one can visually explain why best practices exist by illustrate them with counter-examples. For instance, failed labeling shows some clutters that impair readability, and thus illustrates why it should be carefully designed. It will also give confidence to novices by showing that everyone struggles with design and programming, even experts. This is best said by the following Tweet from our collection of glitches *"makes me feel so much better about my own experiences with d3"*.

Our exploratory work provides a first glance to understand the type of errors that could be interpreted as glitches, but their reverse engineering remains difficult and needs more research. We call for an effort to better support the identification and reflection on glitches during design and programming phases. In our collection we had a Tweet saying *"I don't even know how to reproduce it"*. Identifying and tracking glitches provenance could provide many benefits. First, to better be able to dissect, reflect and understand them. Second, to contextualize them as they usually are only an intermediate step that is interesting if the end goal is known. Glitches could even be used as bookmarks of an interesting step that eventually was pivotal in the final design decisions.

Further work will be needed to systematically collect, structure and expand glitches collections. This will allow assessing how representative the collection we used is of the whole glitches design space, and allow to focus on missing areas. While we have only used two structured collections, we could expand our glitches portfolio by crawling the web using keywords such as glitches *errors, mistakes*, and then by using errors synonyms *unintentional, accidental, aleatory* to cast a broader net. Also, while we have seen animated glitches we have discarded them from our exploratory analysis, but they provide very interesting and captivating behaviors.

There are some remaining questions we still wonder and would like to discuss during the conference: why glitches play with our natural curiosity to explore visual stimuli? Why and how people engage in (not) sharing their failure? Why people find beauty in mistakes? What happens in this space when glitching is an intentional technique rather than an accident? Can glitching provide a way for critical visualization to emerge? Could it be used to create "an unexpected moment in a system that calls attention to that system, and perhaps even leads us to notice aspects of that system that might otherwise go unnoticed"[17]?

Acknowledgements

We would like to thank the inititors of both the accidental-art Tumblr (@kara_woo and @ErikaMudrak) and the #d3brokeandmadeart hashtag on Twitter (@Elijah_Meeks) as well as the contributors to both of those excellent publicly available collections of glitches. We also thank the reviewers for their insightful feedback and references on glitch art during the review process.

References

[1] The art of glitch. *Pbs documentary*, 2012.
[2] The free dictionary. *Glitches*, 2016.
[3] H. Bohnacker, B. Gro\s s, J. Laub, and C. Lazzeroni. Generative gestaltung. *Verlag Hermann Schmidt*, 4, 2009.
[4] M. Bostock, V. Ogievetsky, and J. Heer. D3; Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.
[5] N. Briz. Glitch art historie [s].
[6] W. Brown and M. Kutty. Datamoshing and the emergence of digital complexity from digital chaos. *Convergence: The International Journal of Research into New Media Technologies*, page 1354856511433683, 2012.
[7] T. Dasu. Data Glitches: Monsters in Your Data. In S. Sadiq, editor, *Handbook of Data Quality*, pages 163–178. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-36257-6_8.
[8] E. den Heijer. Evolving glitch art. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 109–120. Springer, 2013.
[9] J.-D. Fekete and C. Plaisant. Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 512–519, New York, NY, USA, 1999. ACM.
[10] D. Geere. Glitch art created by'databending'. *Wired Magazine*, page 118, 2010.
[11] M. Germen. Inadvertent-ars accidentalis. 2008.
[12] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 2013.
[13] S. Huron, R. Vuillemot, and J. D. Fekete. Visual Sedimentation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2446–2455, Dec. 2013.
[14] Y. Jansen and P. Dragicevic. An Interaction Model for Visualizations Beyond The Desktop. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2396–2405, Dec. 2013.
[15] Y. Jansen, P. Dragicevic, P. Isenberg, J. Alexander, A. Karnik, J. Kildal, S. Subramanian, and K. Hornbæk. Opportunities and challenges for data physicalization. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3227–3236. ACM, 2015.
[16] Jodi. World wide wrong. *http://nimk.nl/eng/world-wide-wrong*, 2005.
[17] M. Klee. The long, twisted history of glitch art. *http://kernelmag.dailydot.com/issue-sections/features-issue-sections/12265/glitch-art-history/*, 2015.
[18] R. Kosara. *The US ZIPScribble Map*. 2006.
[19] R. Kosara. Visualization Criticism - The Missing Link Between Information Visualization and Art. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 631–636, July 2007.
[20] H. S. Manon and D. Temkin. Notes on glitch. *world picture*, 6:118, 2011.
[21] S. Mason. Glitched lit: possibilities for databending literature. In *Proceedings of the 2nd workshop on Narrative and hypertext*, pages 41–44. ACM, 2012.
[22] R. Menkman. *The glitch moment (um)*. Institute of Network Cultures Amsterdam, 2011.
[23] R. Menkman. Glitch studies manifesto. *Video vortex reader II: Moving images beyond YouTube*, pages 336–347, 2011.
[24] Mike Bostock. Design is a Search Problem. *OpenVis Conference*, 2014.
[25] Mozilla Developer Network. SVG Attribute reference. 2016.
[26] S. M. Park. *Tall Building Form Generation by Parametric Design Process*. Illinois Institute of Technology, 2005. Google-Books-ID: YSAxHQAACAAJ.
[27] R. M. Roberts. Serendipity: Accidental discoveries in science. *Serendipity: Accidental Discoveries in Science, by Royston M. Roberts, pp. 288. ISBN 0-471-60203-5. Wiley-VCH, June 1989.*, 1, 1989.
[28] R. C. Team and others. R: A language and environment for statistical computing. 2013.
[29] R. Woodbury. Elements of parametric design. 2010.